Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application, where added material is shown in underlined type, deleted material is shown in ~~strikeout type~~:

Listing of Claims:

1. (Previously presented) A method for bandwidth scheduling in a switch comprising a switching fabric, and a bandwidth scheduler located before any queue of said switch, the method comprising the steps of:

receiving a stream of data from the switching fabric;

subjecting the stream to a decision making algorithm in the bandwidth scheduler resulting in that the stream is before said stream enters any queue of said switch.

2. (Original) A method for bandwidth scheduling according to claim 1, wherein the stream of data includes identifiable data packets:

subjecting each data packet to a decision making algorithm in the bandwidth scheduler resulting in that the data packet is accepted or rejected.

3. (Currently amended) A method for bandwidth scheduling according to claim 2, wherein each packet contains information about its flow identity, namely port, identified by port number, ~~(number)~~ and traffic class.

4. (Original) A method for bandwidth scheduling according to claim 3, wherein a limit (BWP_{max}) is set on the maximum accepted bandwidth per port.

5. (Original) A method for bandwidth scheduling according to claim 4, wherein a virtual queue is associated with each port by means of a counter VQLP (virtual queue length of port) and the counter VQLP is increased with the packet length for each accepted packet and updated by subtracting the configuration parameter BWP_{max} (maximum accepted bandwidth per port) each time unit.

6. (Original) A method for bandwidth scheduling according to claim 5, wherein, if the counter VQLP < a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet.

7. (Previously presented) A method for bandwidth scheduling according to claim 4, wherein a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class.

8. (Original) A method for bandwidth scheduling according to claim 7, wherein a virtual queue is associated with each traffic class by means of a counter VQLTC (virtual queue length per traffic class) and the counter VQLTC is increased with the packet length for each accepted packet and updated each time unit by subtracting the configuration parameter $BWTC_{max}$.

9. (Original) A method for bandwidth scheduling according to claim 8, wherein, if the counter VQLTC < a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet.

10. (Original) A method for bandwidth scheduling according to claim 3, wherein a counter TC (traffic class) is increased with the packet length when the traffic class accepts a packet, and a variable TCP_{max} (traffic class port maximum) is set to a value equal to the maximum of the TC counters for the port in question, wherein, for a traffic class having the ratio TC/TCP_{max}< a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet, whereby the bandwidth is distributed in accordance with the Max-Min algorithm.

11. (Original) A method for bandwidth scheduling according to claim 3, wherein each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$).

12. (Original) A method for bandwidth scheduling according to claim 11, wherein a counter TC (traffic class) is increased with the packet length when the traffic class accepts a packet and is updated each time unit by subtracting the configuration parameter $BWTC_{min}$ (bandwidth per traffic class minimum).

13. (Original) A method for bandwidth scheduling according to claim 10, wherein a weight WTC (weight traffic class) is associated with each traffic class, so that the algorithm automatically prioritizes certain traffic classes.

14. (Original) A method for bandwidth scheduling according to claim 13, wherein the counter TC (traffic class) is increased with the packet length multiplied by the configuration parameter WTC (weight traffic class), when the traffic class accepts a packet, and the counter TC is updated each time unit by subtracting a configuration parameter $BWTC_{min}$ (bandwidth per traffic class minimum) multiplied by the configuration parameter WTC.

15. (Original) A method for bandwidth scheduling according to claim 3, wherein, for each traffic class, a counter BL (backlogging) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class.

16. (Original) A method for bandwidth scheduling according to claim 15, wherein a variable BLP_{max} (backlogging port max) stores the maximum of the backlogging counters for the traffic classes of each port, and, for a traffic class with the ratio $BL/BLP_{max} <$ a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet.

17. (Original) A method for bandwidth scheduling according, to claim 16, wherein the BL counter is increased with the packet length multiplied by a configuration parameter WTC (weight traffic class), when the traffic class accepts a packet, and said counter BL is limited to a fixed upper value and a fixed lower value, and if one backlogging counter for a traffic class reaches the upper limit, all other counters are decreased in order to maintain the internal difference, and if one backlogging counter for a flow reaches the lower limit, this counter remains at the lower limit and the counter BL is updated each time unit by subtracting a configuration parameter BWT_{min} (minimum bandwidth per traffic class) multiplied by the configuration parameter WTC.

18. (Original) A method for bandwidth scheduling according to claim 3, wherein, if one traffic class is particularly aggressive or active, it is forced to give up a part of its accepted bandwidth.

19. (Original) A method for bandwidth scheduling according to claim 18, wherein, when a packet is accepted in a traffic class having the maximum accepted bandwidth (TC = TCP_{max}), a charity function forces the packet to be discarded and a counter charity (CH) is increased with a configurable fraction of the accepted packet length (+packet length x give factor), and when a packet is rejected in one of the other traffic classes, the charity function forces the packet to be accepted and the clarity counter (CH) is decreased with the packet length multiplied with the weight of the respective traffic class (-packet length x WTC), and the counter (CH) is updated each time unit by multiplication with a decay factor.

20. (Original) A method for bandwidth scheduling according to claim 3, wherein:

a limit (BWP_{max}) is set on the maximum accepted bandwidth per port, a virtual queue is associated with each port, and a flag is set in dependence of the port queue length;

a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class, a virtual queue is associated with each traffic class, and a flag is set in dependence of the traffic class queue length;

the bandwidth is distributed in accordance with the Max-Min algorithm;

each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$);

a weight (WTC) is associated with each traffic class, so shat the algorithm automatically prioritizes certain traffic classes;

for each traffic class, a backlogging counter (BL) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class;

if one traffic class is particularly aggressive or active, it gives up a part of its accepted bandwidth.

21. (Original) A method for bandwidth scheduling according to claim 3, wherein flows are grouped together by means of a hash function into a set of flow groups.

22. (Original) A method for bandwidth scheduling according to claim 21, wherein a counter FG (flow group) is increased with the packet length when the flow group accepts a packet,

and a variable FG_{max} (flow group maximum) is set to a value equal to the maximum of the FG counters for the traffic class in question, and wherein, for a flow group having the ratio $FG/FG_{max} <$ a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet, whereby the bandwidth is distributed in accordance with the Max-Min algorithm.

23. (Original) A method for bandwidth scheduling according to claim 22, wherein:

a limit (BWP_{max}) is set on the maximum accepted bandwidth per port, a virtual queue is associated with each port, and a flag is set in dependence of the port queue length;

a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class, a virtual queue is associated with each traffic class, and a flag is set in dependence of the traffic class queue length:

each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$);

a weight (WTC) is associated with each traffic class, so that the algorithm automatically prioritizes certain traffic classes;

for each traffic class, a backlogging counter (BL) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class;

if one traffic class is particularly aggressive or active, it gives up a part of its accepted bandwidth.

24. (Original) A method for bandwidth scheduling according to claim 6, 9, 10, 12, 14, 17, or 19, wherein flows are grouped together by means of a hash function into a set of flow groups, and a counter FG (flow group) is increased with the packet length when the flow group accepts a packet, and a variable FG_{max} (flow group maximum) is set to a value equal to the maximum of the FG counters for the traffic class in question, and wherein, for a flow group having the ratio $FG/FG_{max} <$ a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet, whereby the bandwidth is distributed in accordance with the Max-Min algorithm.

25. (Original) A method for bandwidth scheduling according to claim 23, wherein the flags set by the algorithm logic are used in a decision sequence comprising:

if port is switched off, then reject, otherwise;

if Flow Groups are enabled and Flow Group is fair, then accept, otherwise;

if queue (VQLP, VQLTC) longer than DiscardWanted (= desired maximum length), then reject, otherwise if Flow Groups are enabled and queue (VQLP, VQLTC) longer than DiscardPreferred (= preferred maximum length), and the; most aggressive Flow Group, then reject, otherwise

if Traffic Classes are enabled and Traffic Class is fair, then accept, otherwise;

if queue (VQLP, VQLTC) longer than DiscardPreferred (= preferred maximum length), then reject, otherwise:

accept.

26. (Cancelled)

27. (Cancelled)

28. (Previously presented) An arrangement for bandwidth scheduling in a switch comprising a switching fabric, and a bandwidth scheduler located before any queue of said switch the arrangement comprising:

means for receiving a stream of data from the switching fabric;

means for subjecting the stream to a decision making algorithm in the bandwidth scheduler resulting in that the stream is accepted or rejected before said stream enters any queue of said switch.

29. (Original) An arrangement for bandwidth scheduling according to claim 28, wherein the stream of data includes identifiable data packets; and further comprising

means for subjecting each data packet to a decision making algorithm in the bandwidth scheduler resulting in that the data packet is accepted or rejected.

30. (Currently amended) An arrangement for bandwidth scheduling according to claim 29, wherein each packet contains information about its flow identity, namely port, identified by port number, ~~(number)~~ and traffic class.

31. (Original) An arrangement for bandwidth scheduling according to claim 30, wherein a limit (BWP_{max}) is set on the maximum accepted bandwidth per port.

32. (Original) An arrangement for bandwidth scheduling according to claim 31, wherein a virtual queue is associated with each port by means of a counter VQLP (virtual queue length of port) and the counter VQLP is increased with the packet length for each accepted packet and updated by subtracting the configuration parameter BWP_{max} (maximum accepted bandwidth per port) each time unit.

33. (Original) An arrangement for bandwidth scheduling according to claim 32, wherein, if the counter VQLP < a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet.

34. (Original) An arrangement for bandwidth scheduling according to claim 31, wherein a limit ($BWTC_{max}$) is set on the minimum accepted bandwidth per traffic class.

35. (Original) An arrangement for bandwidth scheduling according to claim 34, wherein a virtual queue is associated with each traffic class by means of a counter VQLTC (virtual queue length per traffic class) and the counter VQLTC is increased with the packet length for each accepted packet and updated each time unit by subtracting the configuration parameter $BWTC_{max.}$

36. (Original) An arrangement for bandwidth scheduling according to claim 35, wherein, if the counter VQLTC < a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet.

37. (Original) An arrangement for bandwidth scheduling according to claim 30, wherein a counter TC (traffic class) is increased with the packet length when the traffic class accepts a packet, and a variable TCP_{max} (traffic class port maximum) is set to a value equal to the maximum of the TC counters for the port in question, and wherein, for a traffic class having the ratio TC/TCP_{max} < a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet, whereby the bandwidth is distributed in accordance with the Max-Min algorithm.

38. (Original) An arrangement for bandwidth scheduling according to claim 30, wherein each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$).

39. (Original) An arrangement for bandwidth scheduling claim 38, wherein a counter TC (traffic class) is increased with the packet length when the traffic class accepts a packet and is updated each time unit by subtracting the configuration parameter $BWTC_{min}$ (bandwidth per traffic class minimum).

40. (Original) An arrangement for bandwidth scheduling according to claim 37, wherein a weight WTC (weight traffic class) is associated with each traffic class, so that the algorithm automatically prioritises certain traffic classes.

41. (Original) An arrangement for bandwidth scheduling according to claim 40, wherein the counter TC (traffic class) is increased with the packet length multiplied by the configuration parameter WTC (weight traffic class), when the traffic class accepts a packet, and the counter TC is updated each time unit by subtracting a configuration parameter $BWTC_{min}$ (bandwidth per traffic class minimum) multiplied by the configuration parameter WTC.

42. (Original) An arrangement for bandwidth scheduling according to claim 30, wherein, for each traffic class, a counter BL (backlogging) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class.

43. (Original) An arrangement for bandwidth scheduling according to claim 42, wherein a variable BLP_{max} (backlogging port max) stores the maximum of the backlogging counters for the traffic classes of each port, and, for a traffic class with the ratio $BL/BLP_{max} <$ a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet.

44. (Original) An arrangement for bandwidth scheduling according to claim 43, wherein the counter BL is increased with the packet length multiplied by a configuration parameter WTC (weight traffic class), when the traffic class accepts a packet, and said counter BL is limited to a fixed upper value and a fixed lower value, and if one backlogging counter for a traffic class reaches the upper limit, all other counters are decreased in order to maintain the internal difference, and if one backlogging counter for a flow reaches the lower limit, this counter will remain at the lower

limit and the counter BL is updated each time unit by subtracting a configuration parameter $BWTC_{min}$ (minimum bandwidth per traffic class) multiplied by the configuration parameter WTC.

45. (Original) An arrangement for bandwidth scheduling according to claim 30, wherein, if one traffic class is particularly aggressive or active, it is forced to give up a part of its accepted bandwidth.

46. (Original) An arrangement for bandwidth scheduling according to claim 45, wherein when a packet is accepted in a traffic class having the maximum accepted bandwidth (TC = TCP_{max}), a charity function forces the packet to be discarded and a counter charity (CH) is increased with a configurable fraction of the accepted packet length (+packet length x give factor), and when a packet is rejected in one of the other traffic classes, the charity function forces the packet to be accepted and the charity counter (CH) is decreased with the packet length multiplied with the weight of the respective traffic class (-packet length x WTC), and the counter (CH) is updated each time unit by multiplication with a decay factor.

47. (Original) An arrangement for bandwidth scheduling according to claim 30, wherein:

a limit (BWP_{max}) is set on the maximum accepted bandwidth per port, a virtual queue is associated with each port, and a flag is set in dependence of the port queue length;

a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class, a virtual queue is associated with each traffic class, and a flag is set in dependence of the traffic class queue length;

the bandwidth is distributed in accordance with the Max-Min algorithm;

each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$);

a weight (WTC) is associated with each traffic class, so that the algorithm automatically prioritises certain traffic classes;

for each traffic class, a backlogging counter (BL) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class;

if one traffic class is particularly aggressive or active, it gives up a part of its accepted bandwidth.

48. (Original) An arrangement for bandwidth scheduling according to claim 30, wherein flows are grouped together by means of a hash function into a set of flow groups.

49. (Original) An arrangement for bandwidth scheduling according to claim 48, wherein a counter FG (flow group) is increased with the packet length when the flow group accepts a packet, and a variable FG_{max} (flow group maximum is set to a value equal to the maximum of the FG counters for the traffic class in question, and wherein, for a flow group having the ratio $FG/FC_{max} <$ a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet, whereby the bandwidth is distributed in accordance with the Max-Min algorithm.

50. (Original) An arrangement for bandwidth scheduling according to claim 49, wherein:

a limit (BWP_{max}) is set on the maximum accepted bandwidth per port, a virtual queue is associated with each port, and a flag is set in dependence of the queue length;

a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class, a virtual queue is associated with each traffic class and a flag is set in dependence of the queue length;

each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$);

a weight (WTC) is associated with each traffic class, so that the algorithm automatically prioritizes certain traffic classes;

for each traffic class, a backlogging counter (EL) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class;

if one traffic class is particularly aggressive or active, it gives up a part of its accepted bandwidth.

51. (Original) An arrangement for bandwidth scheduling according to claim 33, 36, 37, 39, 41, 44, or 46, wherein flows are grouped together by means of a hash function into a set of flow groups, and a counter FG (flow group) is increased with the packet length when the flow group accepts a packet, and a variable FG_{max} (flow group maximum) is set to a value equal to the maximum of the FG counters for the traffic class in question, and wherein, for a flow group having

the ratio $FG/FG_{max} <$ a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet, whereby the bandwidth is distributed in accordance with the Max-Min algorithm.

52. (Original) An arrangement for bandwidth scheduling according to claim 50, wherein the flags set by the algorithm logic are used in a decision sequence comprising:

if port is switched off, then reject, otherwise;

if Flow Groups are enabled and Flow Group is fair, then accept, otherwise;

if queue (VQLP, VQLTC) longer than DiscardWanted (= desired maximum length), then reject, otherwise

if Flow Groups arc enabled and queue (VQLP, VQLTC) longer than

DiscardPreferred (= preferred maximum length), and the most aggressive Flow Group, then reject, otherwise

if Traffic Classes are enabled and Traffic Class is fair, then accept, otherwise;

if queue (VQLP, VQLTC) longer than DiscardPreferred (= preferred maximum length), then reject, otherwise;

accept.

53 (Cancelled)

54. (Cancelled)